VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Analýza a vyhodnocení jízdy motorkáře pomocí akcelerometrických dat

Analysis and Evaluation of Motorcyclists Drive using Accelerometer Data

2015/2016 Filip Krupa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Filip Krupa**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2612R041 Řídicí a informační systémy

Téma: Analýza a vyhodnocení jízdy motorkáře pomocí akcelerometrických dat
Analysis and Evaluation of Motorcyclists Drive using Accelerometer Data

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Popis technologie měření zrychlení a přehled výrobců technických řešení.
2. Zpracování principů digitální filtrace signálů.
3. Analýza možností implementace digitálních filtrů v MCU.
4. Zpracování naměřených dat z akcelerometru.
5. Návrh vyhodnocení jízdy motorkáře.
6. Testování navrženého řešení.
7. Zhodnocení výsledků práce.

Seznam doporučené odborné literatury:

[1] VAN SICKLE, Ted. *Programming microcontrollers in C.* 2nd ed. New York: Newnes, c2003, xvi, 454 s. ISBN 1-878707-57-4.
[2] HRDINA, Zdeněk a František VEJRAŽKA. *Signály a soustavy.* Vyd. 1. Praha: ČVUT, 2001 dotisk. 234 s. ISBN 80-010-1726-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Prauzek, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

.....Krupa.....

Filip Krupa

Datum odevzdání bakalářské práce 28. 4. 2016

Poděkování

Děkuji vedoucímu práce Ing. Michalu Prauzkovi, Ph.D. za trpělivost a odbornou pomoc, dále taky své rodině za podporu při vypracování mé práce.

Abstrakt

Cílem mé bakalářské práce je analyzovat a vyhodnotit jízdu motorkáře za pomoci dat z akcelerometru. Abych mohl úspěšně analyzovat tato data, musím se zabývat různými typy akcelerometrů a taky největšími výrobci těchto součástek. Dále pak je potřeba nahlédnout do problematiky mikrokontroléru, ze kterého jsem data pro zpracování získal, data je potřeba zpracovat filtrem, a proto se v této práci zabývám také jeho návrhem a následnou aplikací. Posledním úkolem je zhodnocení výsledků a zvolení správného filtru.

Klíčová slova

akcelerometr, MEMS, mikrokontrolér, Matlab, filtrace, FIR

Abstract

The aim of my thesis is to analyze and evaluate the biker‘s ride by using data from the accelerometer. To be able to successfully analyze these data, I have to deal with different types of accelerometers and also with the largest manufacturers of these components . Furthermore, it is necessary to look into the problems of the microcontroller, from which I gained the data. The data needs to be processed through a filter and thus this work also deals with the design of the filter and its followin applications. The final task is to evaluate the results and select the proper filter.

Key Words

accelerometer , MEMS , microcontroller , Matlab , filtration, FIR

Obsah

Seznam použitých symbolů a zkratek

ALU	aritmeticko-logická jednotka
FIR	Finite Impulse Response
LGA	Land Grid Array
MCU	mikrokontrolér
MEMS	Micro Electro Mechanical Systém
ODR	Output Data Rates
RoHS	Restriction of the use of certain Hazardous Substances

Seznam ilustrací

Seznam tabulek

1 Úvod

Akcelerometry jsou dnes hodně využívané, především v mobilních telefonech a další spotřební elektronice, kde jsme si na jejich přítomnost již zvykli. Za velké rozšíření akcelerometrů mohou především jejich stále se zlepšující parametry. Některé akcelerometry mohou mít programovatelné vstupy a umí reagovat na naprogramované události, dále komunikace přes sběrnici je zcela běžnou a možnost uspání akcelerometrů umožňuje používat je i při napájení z malých baterií.

I přes kvality akcelerometrů je potřeba jejich data zpracovat, k tomuto účelu se nejlépe hodí mikrokontrolér. Mikrokontroléry se často používají pro zpracování jak po stránce filtrace a následné analýzy, tak i pro řízení následné reakce.

Analýza jízdy motorkáře má svá specifika, neboť motorka při jízdě dosahuje sama o sobě velkého přetížení a tyto stavy se mohou velmi lehce přiblížit stavu nehody. Pokud ovšem chceme spolehlivě detekovat jen nehodu, je potřeba data vyfiltrovat a aplikovat na ně vhodné podmínky vyhodnocení.

Digitální filtry jsou velice rozšířené a jejich návrh, ač matematicky složitý, se dá jednoduše realizovat pomocí mnoha dostupných programů. Pro návrh filtrů s konečnou impulzní odezvou je známo mnoho metod, ovšem ne každá se hodí pro návrh všech typů filtru, a taky ne všechny typy filtrů se hodí na veškerá data. Proto je potřeba při návrhu takovýchto filtrů vyzkoušet několik možných variant a zvolit tu správnou.

Výsledkem je filtr vhodný pro zpracování dat z akcelerometru pro účel vyhodnocení jízdy motorkáře a detekce případné nehody.

2 Měření zrychlení

Zrychlení je vektorová fyzikální veličina, která charakterizuje změnu rychlosti za jednotku času. Zrychlení *a* se nejčastěji počítá podle vztahu:

$$a = \frac{dv}{dt} = \frac{d^2s}{d^2t} \tag{1}$$

V mé práci se budu zabývat měřením absolutních vibrací, což je měření pohybu tělesa vzhledem ke gravitačnímu poli země. Výsledné hodnoty měření jsou potom udávány v násobcích tíhového zrychlení.

2.1.1 Tíhové zrychlení

Tíhové zrychlení je zrychlení, které působí na veškeré předměty na Zemi. Toto zrychlení se značí *g* a jeho hodnota je různá pro každé místo na Zemi. Hodnota $\mathrm{g} = 9.8\ \mathrm{m} \cdot \mathrm{s}^{-2}$ má dostačující přesnost pro většinu výpočtů.

2.2 Akcelerometr

Akcelerometr je senzor pro měření zrychlení, který využívá přeměny zrychlení *a* na sílu *F* pomocí tělesa s hmotností *m*. Toto těleso působí na citlivý prvek a mění jeho stav napjatosti (deformaci), elektrické vlastnosti, nebo kompenzuje sílu působící na citlivý prvek. Citlivý prvek je obvykle membrána, vetknutý nosník, struna nebo pružina.

Měření deformace se obvykle provádí tenzometry (polovodičovými nebo kovovými), u deformace pružiny lze využít i změny kapacity, nebo tuto deformaci převést na posun jádra diferenciální indukčnosti. Změna elektrických vlastností se obvykle projeví vytvořením náboje na piezoelektrickém prvku, nebo změnou kapacity.

Snímače jsou omezeny vlastní rezonanční frekvencí, měřené vibrace musí mít výrazně menší frekvenci, než je rezonanční frekvence snímače.

Z principu funkce akcelerometru vyplývá, že je schopen měřit zrychlení jen v jedné ose. Z tohoto důvodu se často používají tříosé akcelerometry, které měří zrychlení ve všech třech osách prostoru. Akcelerometry mohou být pasivní nebo aktivní, podle toho jakým způsobem je vyhodnocována změna stavu citlivého prvku.

2.2.1 Části akcelerometru

- **dotyková část** zajišťuje spojení snímače s objektem
- **seismická hmota** zajišťuje vytvoření síly *F* působící na citlivý prvek. Čím větší je hmotnost *m,* tím citlivější je snímač, neboť $\mathrm{F} = \mathrm{m} \cdot \mathrm{a}$. Zvyšování hmotnosti má za následek snížení rezonanční frekvence snímače, a tím i omezení jeho rozsahu.
- **citlivý prvek** je nejčastěji pružina nebo nosník a senzory vyhodnocující pohyb.

Jednotlivé části akcelerometru jsou umístěny tak, aby byla zajištěna maximální citlivost v požadované ose a potlačena citlivost v jiných osách.

Pro získání správných údajů je potřeba akcelerometr vhodně upevnit. Pro získání nejlepších výsledků se používá šroubový spoj, ale akcelerometr je možné upevnit také pomocí lepení, vosku nebo magnetu. Tyto způsoby ovšem výrazně zhoršují frekvenční charakteristiku. V určitých případech je možné akcelerometr jen přiložit na měřený předmět, potom je ale výsledek silně nepřesný a toto řešení lze použít pro měřené frekvence kmitů menší než 1kHz.

2.2.2 Základní vlastnosti akcelerometrů

Maximální amplituda

Základní parametr při výběru akcelerometru je maximální amplituda, kterou zvládne daný akcelerometr změřit. Nejčastěji se udává v násobcích g.

Dynamický rozsah

Maximální amplituda, kterou lze změřit, než se snímač poškodí.

Frekvenční odezva

Je dána rezonanční frekvencí snímače a jeho hmotností. Je to frekvenční rozsah, při kterém mají výstupní hodnoty definovanou přesnost.

Uzemnění

Jsou dva typy snímačů. První má kryt odizolovaný od okolí a je náchylný na indukované rušení. Druhý má jeden pól spojený s krytem snímače, a proto je potřeba na to brát ohled při jeho upevnění.

Horní frekvenční limit

Maximální frekvence měřeného signálu, kdy se měřená hodnota neliší od pravé víc, než je dovolená odchylka.

Dolní prahová frekvence

Minimální frekvence, při níž ještě měřený signál odpovídá pravému signálu nebo je v povolené odchylce.

Citlivost

Výstupní napětí akcelerometru při měření určité síly vyjádřené tíhovým zrychlením *g*. Nejčastěji 10 mV/g nebo 100 mV/g.

Teplotní vliv

Akcelerometry mají teplotní závislost, a proto je nutné ji kompenzovat. Výrobce často dodává křivky teplotní závislosti, nebo je teplotní kompenzace provedena přímo na senzoru.

Připojení snímače

Pro získání přesných výsledků je potřeba zajistit, aby signál vycházející z akcelerometru nebyl vystaven rušení. Je potřeba použít kvalitní kabel nebo vhodné umístění snímače na desce.

2.2.3 MEMS akcelerometry

MEMS akcelerometry jsou uzpůsobeny k použití na deskách plošných spojů, jejich vnitřní struktura je sofistikovaná a velikost pohyblivých prvků se pohybuje v desítkách nm. Tyto akcelerometry mají trochu jiné parametry než klasické akcelerometry.

Díky mnohem menší hmotnosti mají větší citlivost, která může být i několikanásobná. Dále je u těchto akcelerometrů důležitá spotřeba, která se obvykle pohybuje ve stovkách µA. Tyto akcelerometry se obvykle vyrábějí jako tříose a mají přepínatelné rozsahy, obvykle ±2g, ±4g, ±8g a ±16g. Napájecí napětí je v rozsahu 2 až 3.6 V, ale u některých typů může být i menší.

2.3 Výrobci akcelerometrů

2.3.1 Omega

Společnost byla založena v roce 1962. Dnes nabízí více než 100 000 produktů pro měření a ovládání teploty, vlhkosti, tlaku, napínání, průtoku, síly, výšky hladiny, pH a vodivosti.

Společnost Omega nabízí především průmyslové akcelerometry, jedno až tříosé. Tyto akcelerometry mají různé typy vývodů, možností uchycení a citlivosti. K akcelerometrům nabízí také široké spektrum doplňků, jako jsou zdroje pro akcelerometry, koaxiální kabely atp. Ovšem v jejich nabídce chybí MEMS akcelerometry.

Omega ACC301

Akcelerometr Omega ACC301 je tříosý snímač ideální pro monitorování vibrací ve třech osách. Akcelerometr má velmi nízký šum a široký rozsah frekvencí až do 10kHz. Váha snímače je pouhých 10 gramů a díky titanové konstrukci je velmi odolný. [1]

Obr. 2-1 Akcelerometr Omega ACC301

2.3.2 Bosch Sensortec

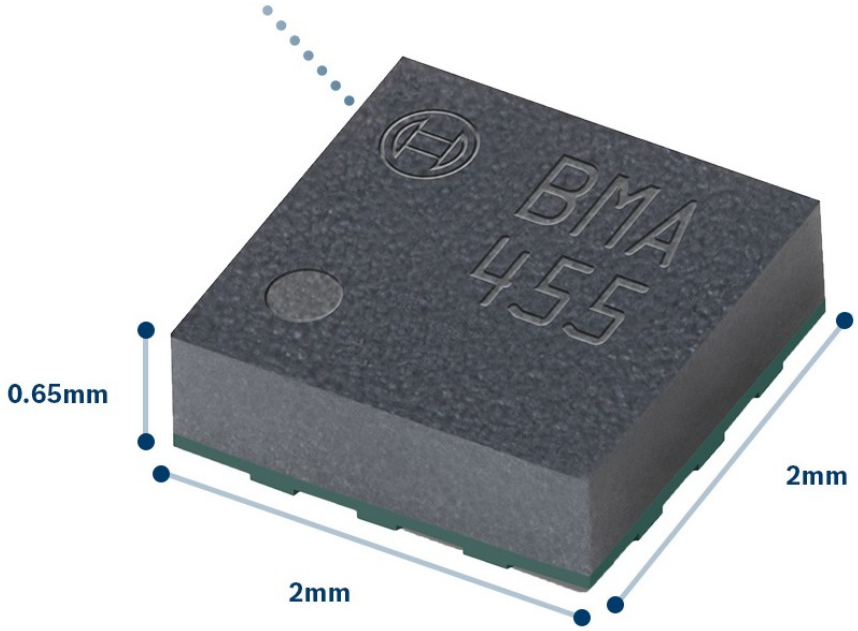
Společnost Bosch Sensortec vyrábí MEMS systémy. Je jednou z největších společností v tomto odvětví. Patří do skupiny Robert Bosch GmbH. Robert Bosch GmbH je německá společnost založena v roce 1886 Robertem Boschem a své sídlo má v německém Gerlingenu. Největší podíl firem ve skupině se věnuje automobilovému průmyslu, ale jsou zde i firmy vyrábějící domácí spotřebiče nebo nářadí.

Bosch Sensortec vyrábí především pohybové senzory, jako například akcelerometry gyroskopy, geomagnetické sensory a kompasy.

Akcelerometry, které tato společnost vyrábí, jsou tříosé, zaměřené na nízkou spotřebu a použití v mobilních telefonech a jiné spotřební elektronice. Pro své akcelerometry požívají LGA obal, což umožňuje jednoduchou aplikaci buďto do socketu, nebo přímo na desku.

BMA455

Velmi malý (zabírá plochu 2 mm^2) tříosý senzor s digitálním výstupem. Má čtrnáctibitové rozlišení a díky své nízké spotřebě je určen pro použití v mobilních telefonech a elektronických doplňcích. [2]



Obr. 2-2 Bosch Sensortec BMA 455

2.3.3 NXP Semiconductors

Společnost sídlí v nizozemském Eidhovenu. Vznikla z divize firmy Philips v roce 2006. V roce 2015 převzala společnost Freescale Semiconduktors. Společnost Freescale se specializovala na výrobu mikrokontrolérů a mikroprocesorů.

Dnes společnost vyrábí velké množství procesorů, mimo jiné i procesory ARM, které jsou často v mobilních telefonech. Dále vyrábí velké množství logických prvků, jako jsou komparátory operační zesilovače stabilizátory, digitální multiplexery atd.

MEMS akcelerometry této společnosti mají přepínatelné rozsahy ±2g, ±4g a ±8g a nízkou spotřebu.

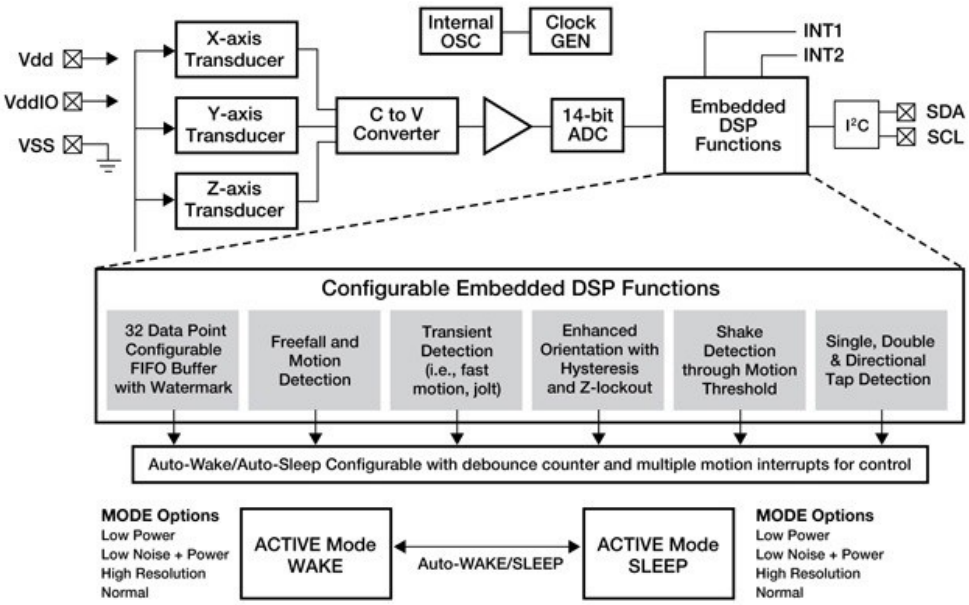
MMA8451Q

Tento akcelerometr byl použit pro získání dat v mé bakalářské práci. Jedná se o čtrnáctibitový digitální akcelerometr a má tyto funkce:

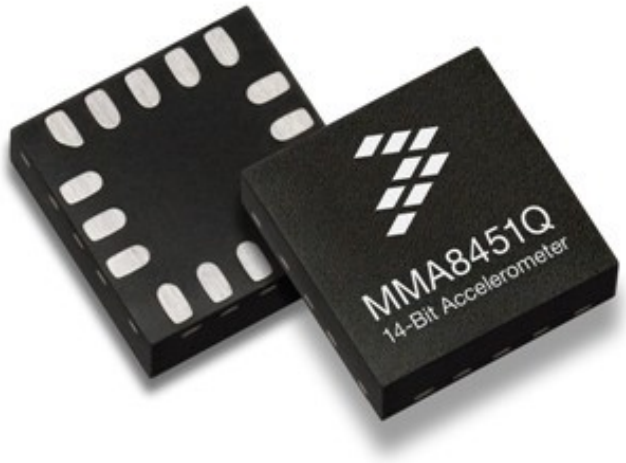
- 1,95 – 3,6 V napájení
- ±2g, ±4g a ±8g dynamicky volitelné rozlišení
- frekvence výstupních dat (ODR) 1,56 Hz – 800 Hz.
- 99 µg/√Hz šum
- 14 a 8 bitový digitální výstup

- výstupní komunikace přes I²C sběrnici (2,25 MHz)
- dva programovatelné piny pro přerušení pro 7 přerušovacích signálů
- tři oddělené kanály pro detekci pohybu:
    - volný pád, pohyb
    - detekce pulzů
    - detekce trhnutí
- detekce orientace (krajina/portrét) s programovatelnou hysterezí
- automatická změna ODR pro probuzení a uspání
- 32 vzorkový FIFO buffer
- filtr typu horní propust na vzorek a celý buffer
- Self-test
- je kompatibilní se směrnicí RoHS
- spotřeba 6 µA - 165 µA

Z vlastností vyplývá, že se jedná o kvalitní akcelerometr s pokročilými funkcemi, který se dá použít v širokém spektru spotřební elektroniky od mobilních telefonů přes čtečky a herní zařízení až po detekci a monitorování vibrací. Veškeré možnosti akcelerometru jsou vidět v blokovém diagramu Obr. 2-3. [3]



Obr. 2-3 Blokový diagram akcelerometru MMA8451Q



Obr. 2-4 NXP Semiconductors MMA8451Q

2.3.4 Analog Devices

Americká firma sídlící v Norwood ve státě Massachusetts. Největší světová firma ve výrobě konvertorů dat. Společnost vyrábí integrované obvody pro analogové a digitální zpracování signálu. Firmu založil Ray Stata a Matthew Lorber v roce 1965. V začátcích firma vyráběla zesilovače.

Co se týče MEMS prvků, tak firma vyrábí akcelerometry, gyroskopy a integrované měřící zařízení. Svými produkty se zaměřují spíše na měření v průmyslu. Jejich akcelerometry mají často velmi nízkou spotřebu.

ADXL350

Tento tříosý akcelerometr má maximální spotřebu 45 μA a rozlišení až 13 bitů. [4]

2.3.5 STMicrocelektronic

Největší evropský výrobce polovodičových čipů. Francouzko-italská firma, která má sídlo ve švýcarské Ženevě. Vznikla v roce 1987 sloučením francouzské a italské firmy zabývající se výrobou polovodičových čipů.

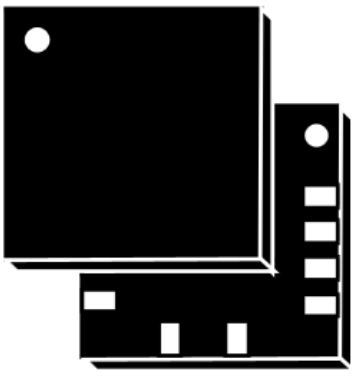
V oblasti MEMS akcelerometru firma vyrábí velké množství akcelerometrů pro automobilový průmysl, a speciálně pro použití v bezpečnostních prvcích. Dále také vyrábí akcelerometrické čipy pro použití v medicíně, které je možné implantovat. Dále nabízejí i analogový MEMS akcelerometr a také akcelerometry s rozlišením až ±400g. Nabídka společnosti v oblasti akcelerometrů je vidět na Obr. 2-5.

<table>
<tr><td rowspan="2">Applications</td><td colspan="4">Package size (mm)</td><td rowspan="2">Features</td></tr>
<tr><td colspan="2">2 x 2 x 1 mm</td><td colspan="2">3 x 3 x 1 mm</td><td>> 4 x 4 x 1.5 mm</td></tr>
<tr><td rowspan="8">Consumer & Industrial</td><td colspan="2"></td><td>12-bit</td><td>H3LIS331DL</td><td rowspan="3"></td><td rowspan="3">High-g</td></tr>
<tr><td colspan="2"></td><td>8-bit</td><td>H3LIS200DL
H3LIS100DL</td></tr>
<tr><td colspan="2"></td><td>12-bit</td><td>LIS331HH</td></tr>
<tr><td>16-bit</td><td>LIS2HH12</td><td colspan="2">LIS3DSH</td><td rowspan="4"></td><td rowspan="4">Low-g</td></tr>
<tr><td>14-bit</td><td>LIS2DS12</td><td colspan="2"></td></tr>
<tr><td>12-bit</td><td>LIS2DH12
LIS2DH</td><td colspan="2">LIS3DH</td></tr>
<tr><td>8-bit</td><td>LIS2DE
LIS2DE12</td><td colspan="2">LIS3DE</td></tr>
<tr><td colspan="2"></td><td colspan="2">LIS344ALH</td><td></td><td>Analog</td></tr>
<tr><td>Long-life applications</td><td colspan="2">IIS2DH</td><td colspan="2"></td><td>IIS328DQ</td><td>10-years commitment</td></tr>
<tr><td>Automotive Non-safety</td><td colspan="2"></td><td colspan="2"></td><td>AIS328DQ
AIS3624DQ</td><td rowspan="2">AEC-Q100 qualified</td></tr>
<tr><td>Automotive Safety (Central & Peripherals Airbags)</td><td colspan="2"></td><td colspan="2"></td><td>AIS1120SX
AIS2120SX
AIS1200PS</td></tr>
<tr><td>Medical</td><td colspan="2">MIS2DH</td><td colspan="2"></td><td></td><td>For Implantable devices</td></tr>
</table>

Obr. 2-5 Nabídka akcelerometrů STMicroelektronic

LIS2DE12

Tříosý akcelerometr s rozlišením ±2g, ±4g, ±8g a ±16g, 8 bitovým výstupem a spotřebou 2 μA. Tento akcelerometr umí vytvořit přerušovací signál a detekovat volný pád. [5]



LGA-12 (2.0x2.0x1 mm)

Obr. 2-6 STMikroelektronic LIS2DE12

3 Digitální filtrace

Díky nárůstu výkonů počítačů i mikrokontrolérů začíná u mnohých aplikací digitální filtrace vytlačovat tu analogovou. Digitální filtrace je sled matematických kroků, díky nimž není naprogramování digitálního filtru velký problém a jeho velkou výhodou je možnost změnit vlastnosti filtru jen změnou několika koeficientů.

Za nevýhodu lze považovat omezení kmitočtu filtru rychlostí prováděných operací. Z toho plyne, že na každém zařízení je omezený maximální kmitočet filtru, a to podle rychlosti procesoru. Tento nedostatek se ještě více projeví u mikrokontrolérů, které mají omezený výpočetní výkon I zde ovšem platí, že digitální filtrace lze použít a s vývojem mikrokontrolérů se používá čím dál častěji.

Digitální filtrace se provádí pomocí číslicových filtrů, které pracují v diskrétním čase, a při jejich realizaci hledáme vhodnou přenosovou funkci, která bude odpovídat našim požadavkům a zároveň bude realizovatelná. Číslicový filtr je popsán v z-rovině přenosovou funkcí, která v časové oblasti odpovídá diferenciální rovnici s konstantními koeficienty. Při realizaci filtru obvykle vycházíme z této diferenciální rovnice. Filtry dělíme na dvě základní skupiny podle délky impulsní odezvy:

- IIR (Infinite Impuls Response)
- FIR (Finit Impulse Response)

3.1 IIR

IIR neboli filtry s nekonečnou impulsní odezvou mají přenosovou funkci ve tvaru:

$$H(z) = \frac{\sum_{j=0}^{M} b_j z^{-j}}{1 + \sum_{i=0}^{N} a_i z^{-i}} \tag{2}$$

kde N je řád filtru, obvykle platí, že M ≤ N. Pro dosažení stability musí póly ležet uvnitř jednotkové kružnice. IIR filtr má vždy zpětnou vazbu, a proto se někdy nazývá rekurzivní.

Filtr IIR nemá zaručenou stabilitu a u tohoto typu číslicových filtrů nelze rovněž dosáhnout lineární fáze, lze se k ní jen přiblížit.

U číslicových filtrů je požadavek na co nejmenší řád filtru a z tohoto hlediska je IIR filtr mnohem výhodnější, neboť je obvyklé, že FIR filtr potřebuje pro dosažení stejné přenosové funkce i desetinásobný řád, což přináší velké nevýhody při jeho realizaci. [6]

3.2 FIR

FIR neboli filtr s konečnou impulzní odezvou má přenosovou funkci ve tvaru:

$$H(z) = \sum_{n=0}^{N} b[n] z^{-n} \tag{3}$$

kde řád filtru je N a délka filtru je N+1. Jak už bylo naznačeno, filtr tohoto typu je vždy stabilní, což je velká výhoda při jeho návrhu. FIR filtr nemá zpětnou vazbu a někdy se nazývá konvoluční. Tyto filtry také mají lineární fázovou charakteristiku. Za nevýhodu se dá považovat již zmíněná nutnost většího řádu oproti IIR. Jelikož v mé bakalářské práci jsem se zabýval filtry FIR, budu se nadále věnovat jen těmto filtrům.

Návrh FIR filtru

Základní typy FIR filtrů jsou stejně jako u klasických analogových filtrů dolní a horní propust a pásmová zádrž a propust.

FIR filtr je popsán diferenciální rovnicí:

$$y[x] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N+1] = \sum_{k=0}^{N-1} b_k x[n-k] \tag{4}$$

kde b_k jsou reálné koeficienty filtru a N délka filtru (řád je N-1).

Často se využívá také zápisu pomocí konvoluce:

$$y[n] = \sum_{k=0}^{N-1} h[k] x[n-k] \tag{5}$$

Tento zápis vyjadřuje konvoluci vzorků impulzní odezvy h[k] a vstupního signálu x[n]. Pro získání frekvenční charakteristiky dosadíme do vztahu (3) $z = e^{j\omega}$:

$$H\left(e^{j\hat{\omega}}\right) = \sum_{n=0}^{N} b[n] e^{-j\hat{\omega}n}, \qquad -\pi < \hat{\omega} < \pi \tag{6}$$

kde $\hat{\omega} = \omega T$ je číslicový uhlový kmitočet. Filtry FIR mohou mít lineární fázi. Potom je frekvenční fázová charakteristika dána vztahem:

$$\Phi(\hat{\omega}) = \arg H(e^{j\hat{\omega}}) = -M\hat{\omega} \tag{7}$$

kde M je konstanta úměrnosti fázové charakteristiky. Skupinové zpoždění je definováno jako záporná derivace fázové charakteristiky:

$$\tau(\hat{\omega}) = -\frac{d\Phi(\hat{\omega})}{d\hat{\omega}} \tag{8}$$

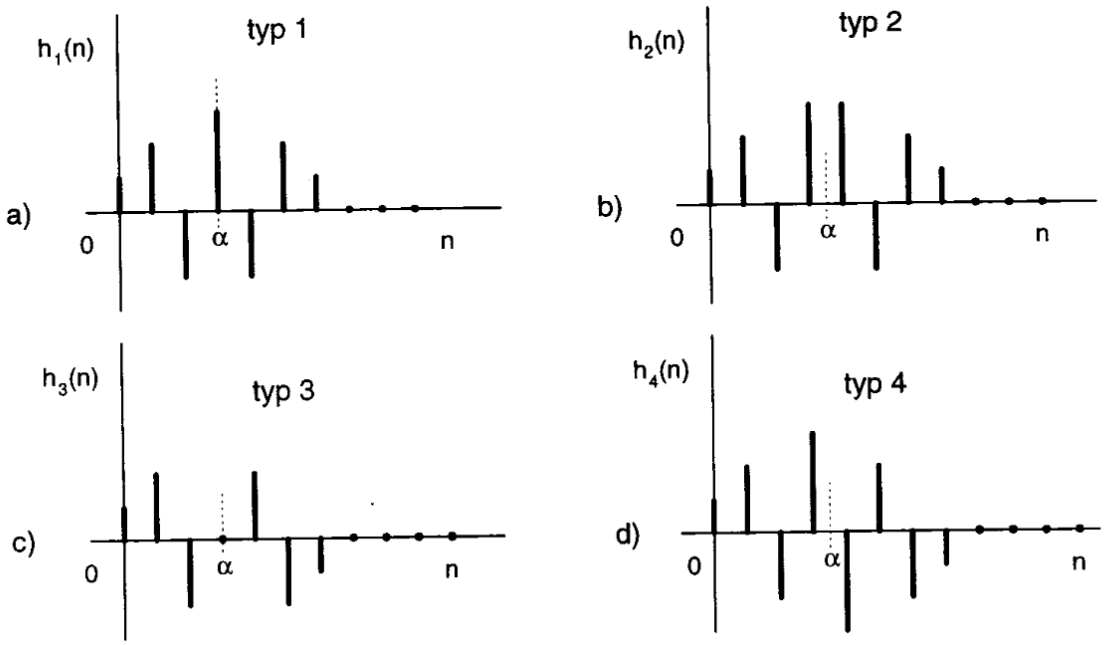
Lineární fáze znamená konstantní skupinové zpoždění a to bude splněno, pokud bude fázová charakteristika lineární funkce kmitočtu $\hat{\omega}$.

$$\Phi(\hat{\omega}) = -\hat{\tau}\hat{\omega} \tag{9}$$

Po upravách viz [6] str 181-182 dospějeme k výrazu:

$$h[n] = h[N-1-n], \hat{\tau} = \frac{N-1}{2} \tag{10}$$

Impulzní odezva je symetrická ke svému středu symetrie. Hovoříme o sudé symetrii $M = (N-1)/2$. Sudá symetrie může mít z hlediska počtu prvků impulzní odezvy 2 typy, a to v případě lichého N se jedná o typ 1 a v případě sudého N se jedná o typ 2 viz Obr. 3-1



Obr. 3-1 Impulzní charakteristiky FIR filtrů s lineární fází

Pro typ 1, kde je lichý počet prvků, například 7 jako na obrázku, je osou symetrie prvek $n = (N-1)/2$. Pro typ 2, který má sudý počet prvků, je osa symetrie mezi prvky $n = (n-2)/2$ a $n = n/2$.

Pokud bude mít fázová charakteristika tvar:

$$\Phi(\hat{\omega}) = \Phi_0 - \hat{\tau}\hat{\omega} \tag{11}$$

kde Φ_0 je konstanta $\Phi_0 = \pm\pi/2$, můžeme obdobně odvodit tvar:

$$h[n] = -h[N-1-n], \hat{\tau} = \frac{N-1}{2} \tag{12}$$

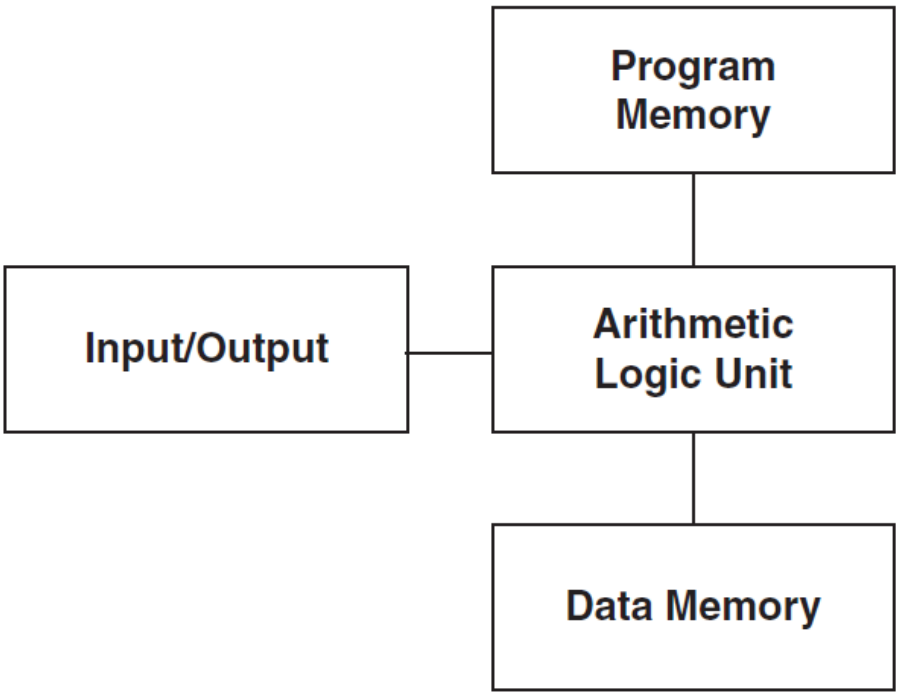
A hovoříme o liché symetrii, která může mít opět 2 typy. Pro typ 3 leží osa symetrie na prvku $n = (N-1)/2$ a N je liché. Pro tento typ musí dále platit, že tento prvek je nulový. Typ 4 má N sudé a osa symetrie leží mezi prvky $n = (n-2)/2$ a $n = n/2$, dále musí platit, že jejich velikosti jsou si rovny.

Filtry s impulzní charakteristikou, která vykazuje lichou symetrii, se používají pouze ve speciálních případech. Typy 1 a 2, které vykazují sudou symetrii, se používají velmi často, ale typ 2 není vhodný pro horní propust a pásmovou zádrž. [7]

4 Implementace digitálních filtrů v MCU

4.1 MCU

MCU neboli mikrokontrolér/mikropočítač si mnoho lidí může splést s mikroprocesorem, který máme všichni ve svém počítači. Je tady ale jeden zásadní rozdíl. Mikroprocesor je pouhá výpočetní jednotka, která potřebuje ke své správné funkci další periferie jako například paměť, sériové a paralelní porty a mnoho dalších. Zatímco MCU je jeden čip, u kterého jsou veškeré tyto periferie integrovány.



Obr. 4-1 Základní blokové schéma MCU

Základní blokový diagram MCU je na Obr. 4-1, kde můžete vidět, že nejdůležitější částí MCU je ALU.

4.1.1 Aritmeticko-logická jednotka

Jedná se o centrální řídící jednotku, která je připojena k bloku vstupů a výstupů a k dvěma paměťovým blokům. Jeden blok je pro paměť programu a druhý pro data.

Počítačová architektura, u které je oddělena paměť programu a dat, se nazývá Harvardská architektura a její velkou výhodou je možnost přistupovat do obou pamětí najednou díky dvěma odděleným sběrnicím.

Většina dnešních počítačů ovšem používá Von Neumannovu architekturu, která je typická tím, že má oba typy paměti sjednoceny do jedné a k přístupu používá jednu sběrnici. To se může zdát jako nevýhoda, ovšem i procesory s touto architekturou mohou dosáhnout několika milionů instrukcí za sekundu. Díky sjednocené paměti a jedné sběrnici je také tento systém jednodušší a především levnější, a proto také používanější.

Vykonávání programu na MCU probíhá tak, že ALU načte instrukci z paměti programu a vykoná ji, poté načte další instrukci a tak pokračuje, dokud program neskončí nebo se nedostane do smyčky, ve které zůstane, dokud nenastane událost, která jí umožní nekonečnou smyčku opustit.

Z nekonečné smyčky se MCU může dostat např. resetem systému, v tom případě se na daný pin přivede daný signál a celé MCU se resetuje, což znamená, že se přepíší určité registry. Další dvě možnosti zahrnují přerušení. ALU kontroluje před vykonáním každé instrukce, jestli nenastalo nějaké přerušení, a pokud nastalo, tak přestane vykonávat hlavní program a vykoná přerušení.

Interní přerušení nastane, pokud některý vnitřní blok, často časovač, vyvolá přerušení. Druhou možností je externí přerušení, které je vyvoláno externím prvkem tak, že přivedeme na daný pin určené napětí. Při výskytu přerušení se začne vykonávat servisní rutina přerušení, což zahrnuje sled událostí řešících dané přerušení, mezi které patří načtení vektoru přerušení a vykonání daných instrukcí. Vektor přerušení ukazuje na místo v paměti, od něhož má ALU začít načítat instrukce pro dané přerušení. Více o problematice fungování MCU je možné najít v [8] kapitola 3.

4.2 Programování filtru v MCU

Nejprve je třeba říct, že MCU se nejčastěji programují v jazyce C. Jazyk C je velice rozšířený a jeho výhodou je lepší čitelnost kódu než při programování v jazyku symbolických adres, a také větší možnosti přenositelnosti kódu mezi procesory.

Pro programování filtrů je vhodné použít MCU, kde bude ALU podporovat operace násobení, neboť pro filtraci je typický velký počet kroků, u kterých se provádí prosté násobení dvou čísel a následné sčítání.

Jádrem FIR filtru je výpočet konvoluce dvou stejně velkých polí:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \tag{13}$$

Při realizaci výpočtů je pole h naplněno koeficienty filtru, které se pro jeden filtr nemění. Druhé pole je potřeba naplnit pro každou sumu jinými daty. Přesněji je potřeba přidat jednu novou hodnotu na straně jedné a na druhé straně jednu hodnotu odebrat, aby byl zachován konstantní počet prvků.

Toto chování přesně popisuje buffer typu FIFO (First In First Out). FIFO je jeden ze základních typů bufferů. Pro představu se dá říct, že se jedná o frontu dat, ve které ten, kdo přijde první, taky první odejde. Pro konvoluci je vhodný právě tento typ bufferu.

Druhý nejčastější typ bufferu je FILO (Firs In Last Out). Pro tento buffer je typické, jak už z názvu vyplývá, že prvek, který do něho vložíme jako první, je vyzvednut jako poslední. Nejlépe se tento buffer představuje jako zásobník.

Pokud chceme filtrovat data v reálném čase, musíme dbát na to, abychom vybrali dostatečně rychlý MCU vzhledem k frekvenci vzorkování našich dat. V mém případě byla vzorkovací frekvence 1,56 Hz, a proto nebyly nároky na MCU nijak velké.

Pokud je filtrace použita pouze jako úprava signálu, u kterého je následně provedena například autokorelace nebo Fourierova transformace, jsou tyto výpočty násobně náročnější než samotná filtrace, především pokud je využito neoptimalizovaných algoritmů.

5 Postup vypracování

5.1 Volba prostředků

Mým úkolem bylo vytvořit aplikaci, která zpracuje data z tříosého akcelerometru vhodně navrhnutým filtrem. Tohoto cíle jsem mohl dosáhnout využitím Matlabu, nebo vytvořit svůj program za použití jazyka C#.

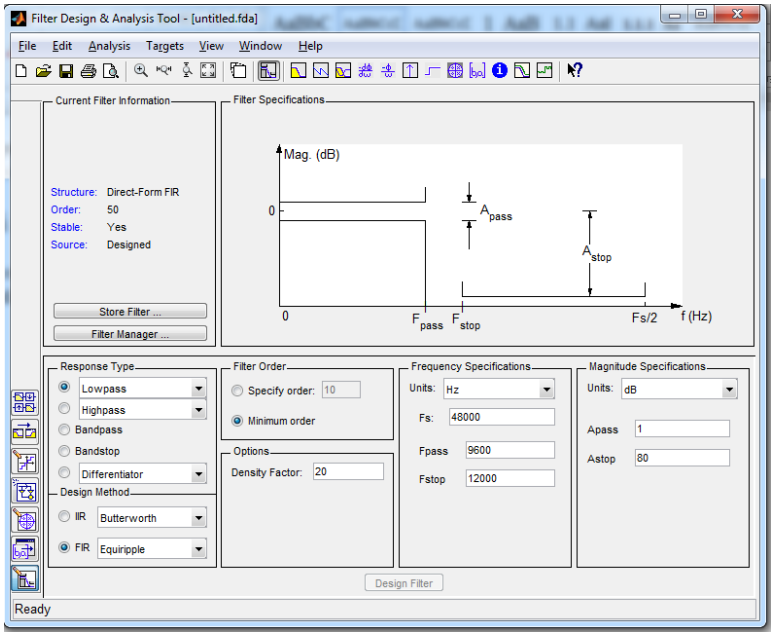
Matlab je skriptovací programovací jazyk, který nabízí pro práci s poli velké množství funkcí a také má rozsáhlou podporu pro návrh a testování číslicových filtrů. Jeho výhodou je také to, že veškeré objekty považuje za matice a umí s nimi velice efektivně pracovat. V mnoha publikacích týkajících se tohoto tématu můžeme najít i kódy pro Matlab, které řeší danou problematiku. Matlab má také nástroj zvaný FDATool (Filter Design and Analysis Tool), který je široce využíván pro návrh filtrů.

Jazyk C# nabízí možnost realizace zcela originální aplikace, ale také spoustu již vytvořených knihoven, které se zabývají filtrací a zpracováním dat. C# je plnohodnotný vysokoúrovňový programovací jazyk vyvinutý společností Microsoft, který je založený na jazyce C a Java. Tento jazyk lze společně s platformou .NET Framework využít pro tvorbu aplikací ve Windows.

Zvolil jsem si jazyk C#, který je mi bližší a nabízí možnost grafické úpravy aplikace. Díky tomu, že byl vyvinut firmou Microsoft, má lepší podporu komunikace s operačním systémem Windows, pro který jsem svou aplikaci tvořil. Pro samotný návrh filtru budu ovšem využívat FDATool, neboť tento nástroj umí výborně navrhnout koeficienty filtru.

Jako vývojové prostředí jsem zvolil Microsoft Visual Studio, které bylo mimo jiné vytvořeno právě pro tvorbu aplikací v jazyce C#, a poskytuje veškerou podporu potřebnou pro kvalitní vývoj aplikace.

Pro úspěšný návrh koeficientů filtru, které jsem potřeboval, bylo potřeba naučit se pracovat s nástrojem FDATool z Matlabu.



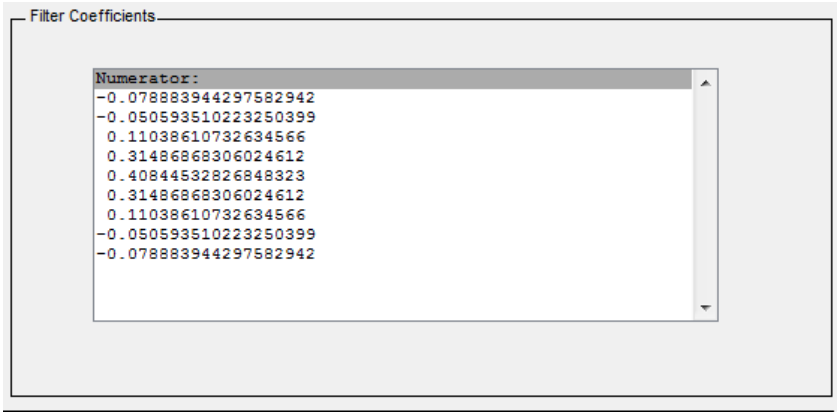
Obr. 5-1 Úvodní okno FDA

5.2 Návrh koeficientů

Při práci v nástroji FDATool je potřeba nejprve zvolit, jestli chci navrhnout filtr FIR nebo IIR, a zvolit metodu návrhu. Pro FIR filtr je přítomno 11 možných metod, já jsem pro první návrh zvolil metodu oken. Pro tuto metodu je možné zvolit řád filtru, já jsem si zvolil 8, a potom typ okna. Typů oken existuje velké množství, já jsem zvolil Kaiserovo okno. Kaiserovo okno vyžaduje také parametr beta, kterým lze upravit šířku přechodového pásma. Beta parametr jsem nechal na předdefinované hodnotě 0,5.

Dále jsem musel zvolit frekvenci vzorkování, která je 1,56 Hz, tato frekvence je nastavena v mikrokontroléru, a tudíž jsem ji nemohl měnit. Dále jsem volil frekvenci řezu, v mém případě 0,3 Hz. FDAtool nabízí i možnost specifikovat frekvenci řezu v intervalu od 0 do 1, kde 1 reprezentuje Nyquistovu frekvenci (polovina vzorkovací frekvence).

Pro navrhnutý filtr můžeme vidět grafy amplitudové a fázové odezvy, skupinové a fázové zpoždění a další, ale pro mě je důležitý výpis koeficientů filtru, který je vidět na Obr. 5-2. Na tomto obrázku lze vidět, že koeficienty jsou vypočítány na velký počet desetinných míst a že pro 8. řád filtru se jedná o 9 koeficientů, což je vlastnost FIR filtru.



Obr. 5-2 Navrhnuté koeficienty filtru v FDATool

Takto získané koeficienty jsem v prvních fázích tvorby své práce ručně přepisoval, později jsem je exportoval do Matlabu nebo přímo do textového souboru a dále s nimi pracoval.

5.3 Tvorba aplikace

5.3.1 Filtr

Samotný algoritmus filtru není nijak složitý, na vstupu je potřeba zadat koeficienty filtru a pole nefiltrovaných dat a na výstupu dostaneme pole dat filtrovaných.

Algoritmus je zobrazen na Obr. 5-3. Na obrázku je vidět, že pro ukládání dat jsem zvolil List. Práce s listem je pomalejší, ale můj program nemá požadavky na rychlost zpracování dat, a proto jsem mohl plně využít výhody listu, jako například snadné přidávání prvků a následné procházení. Dále je algoritmus již upraven tak, aby zvládl filtry různých řádů, první verze mého algoritmu uměla pouze filtry 8. řádu, ovšem velice brzy jsem narazil na potřebu testovat i jiné řády filtru, a proto jsem musel algoritmus upravit.

```
private void filtr(List<double> data, List<double> fltdata, List<double> coef)
{
    double[] polex = new double[coef.Count];
    double sum = 0;

    for (int a = 0; a <= data.Count - (coef.Count+1); a++)
    {
        for (int b = 0; b < coef.Count; b++)
        {
            polex[b] = data[a - b + coef.Count];
        }
        for (int c = 0; c < coef.Count; c++)
        {
            sum += polex[c] * coef[c];
        }
        fltdata.Add(sum);
        sum = 0;
    }
```

Obr. 5-3 Algoritmus FIR filtru

5.3.2 Načítání dat

Pro otevření souboru s daty používám třídu OpenFileDialog, pro načtení dat třídu StreamReader. Cyklicky načítám řádek dat. Data jsou ve formátu .csv, kde jako oddělovač je použita čárka. Každý řádek dat má 5 prvků.

První prvek je pořadové číslo vzorku, to pro mě nemělo žádný význam, a proto jsem ho nikam neukládal a po načtení bylo zahozeno.

Druhý prvek je příznak, viz 5.3.3. Příznaky jsem v prvních fázích vývoje své aplikace také zahazoval, ale v pozdějších verzích se ukázaly jako velice důležité pro vyhodnocení mého filtru, a proto jsem je začal ukládat do listu.

Třetí až pátý prvek jsou data z každé osy akcelerometru v pořadí X, Y, Z. Tato data jsem potřeboval pro svůj filtr, a proto jsem pro každou osu vytvořil jeden list a do těchto objektů jsem data přidával.

Každý datový soubor má jiný počet řádků podle délky testu. Maximální počet řádků mých datových souborů může být 2013. Toto číslo je maximum paměti vyhrazená pro tento účel v mikrokontroléru a odpovídá asi 20 minutám záznamu jízdy. Pokud je jízda delší, tak v paměti zůstane 20 posledních minut.

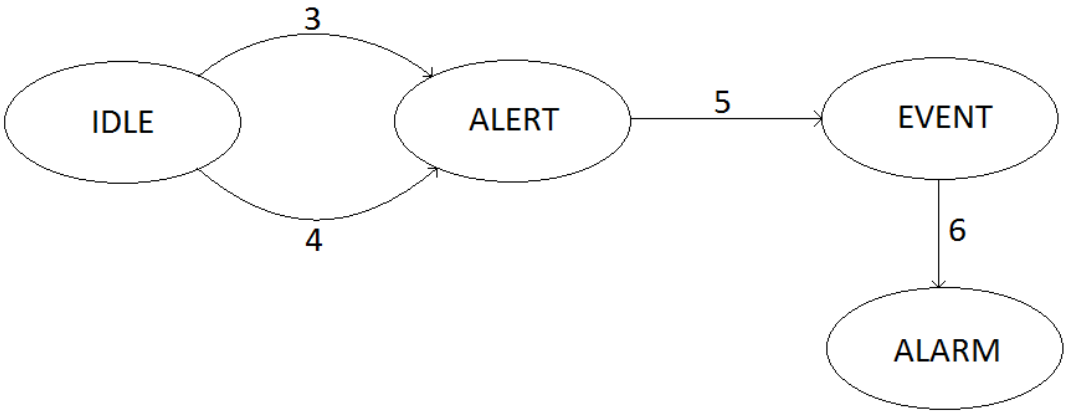
5.3.3 Příznaky a stavy

Příznaky jsou tvořeny mikrokontrolérem z dat, která si vnitřně filtruje svým FIR filtrem řádu 3. Příznaky mohou nabývat hodnot 0 až 7. Program v mikrokontroléru rozlišuje 4 stavy, viz Tab. 5-1.

Tab. 5-1Seznam stavů mikrokontroléru

Idle	Stav, při kterém probíhá měření.
Alert	Stav, kdy je možné, že motorka havarovala.
Event	Stav, kdy je vyhodnoceno, že došlo k nehodě.
Alarm	Stav, kdy je signalizována nehoda motorkáře.

Příznaky 3 až 6 jsou přechody mezi stavy, viz Obr. 5-4. Pro mou práci je důležitý stav Idle a Alert. Stavy Event a Alarm souvisejí s funkcí aplikace na MCU, kdy po detekci havárie je spuštěn časovač (stav EVENT) a po uplynutí definovaného času je odeslán signál o havárii na předem zvolené číslo.



Obr. 5-4 Stavy mikrokontroléru

Příznak 0 se v každém souboru dat objeví jen jednou a indikuje první vzorek.

Příznak 1 se také objevuje jen jednou a indikuje vzorek poslední.

Příznak 2 je u každého vzorku, který je vyhodnocen jako normální a systém se nachází ve stavu Idle.

Příznak 3 informuje o přechodu ze stavu Idle do stavu Alert a nastane, pokud je prostorový vektor zrychlení v intervalu $\langle 0{,}92; 1{,}08 \rangle$ a zároveň je úhel mezi daným vektorem a kalibračním vektorem větší než 50°.

Příznak 4 indikuje přechod zpět do stavu Idle, a to ve chvíli kdy nejsou splněny podmínky pro stav Alert.

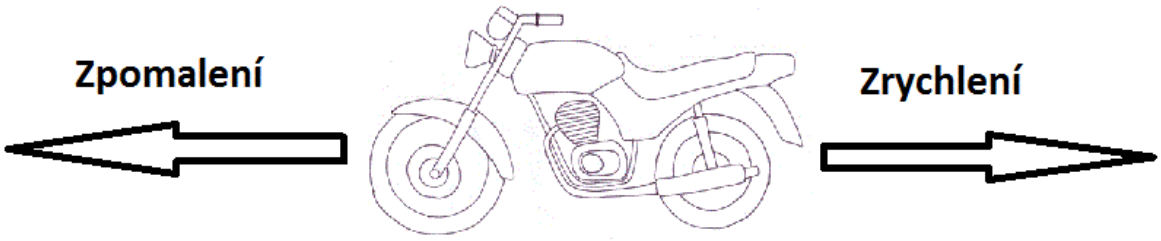
Příznak 5 je pro přechod do stavu Event ve chvíli, kdy je systém nepřerušeně ve stavu Alert po 10 kroků, tedy asi 6,4 s.

Příznak 6 indikuje přechod do stavu Alarm, který nastane po stavu Event a v tomto stavu systém hlásí nehodu.

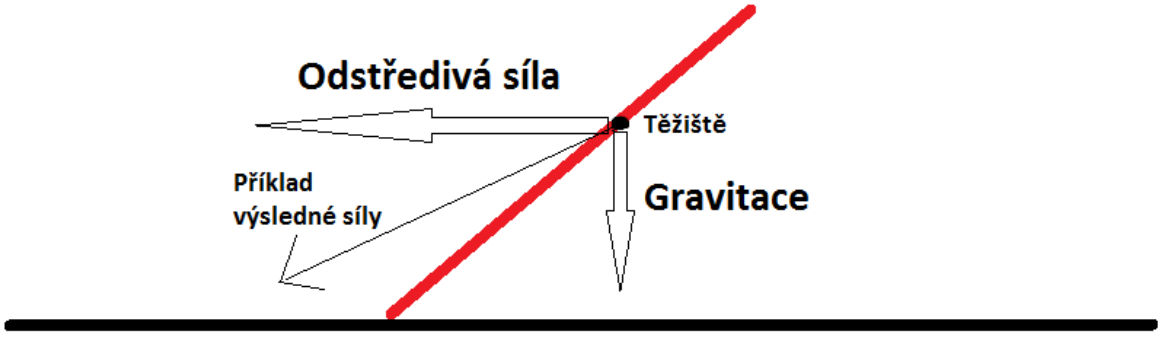
Příznak 7 je u druhého vzorku a označuje data, která byla použita pro kalibraci vyhodnocování, a podle těchto dat se nastavuje kalibrační vektor.

5.3.4 Volba vyhodnocení

MCU se aktivuje při nastartování, a to předpokládá, že motorka neleží na zemi, proto je jako druhý vzorek sejmut kalibrační vektor. Při jízdě generuje tříosý akcelerometr v MCU data, která se dají vyložit jako vektor směru okamžitého zrychlení. Ve chvíli, kdy na motorku nepůsobí žádné síly, směřuje tento vektor zrychlení do středu země, neboť na něho působí gravitační zrychlení. V průběhu jízdy se k tomuto zrychlení přidá ještě zrychlení/zpomalení motorky a také odstředivé/dostředivé síly působící v zatáčkách, viz Obr. 5-5 a Obr. 5-6.



Obr. 5-5 Síly působící na motorku při zrychleni a zpomalení



Obr. 5-6 Síly působící na motorku při průjezdu zatáčkou

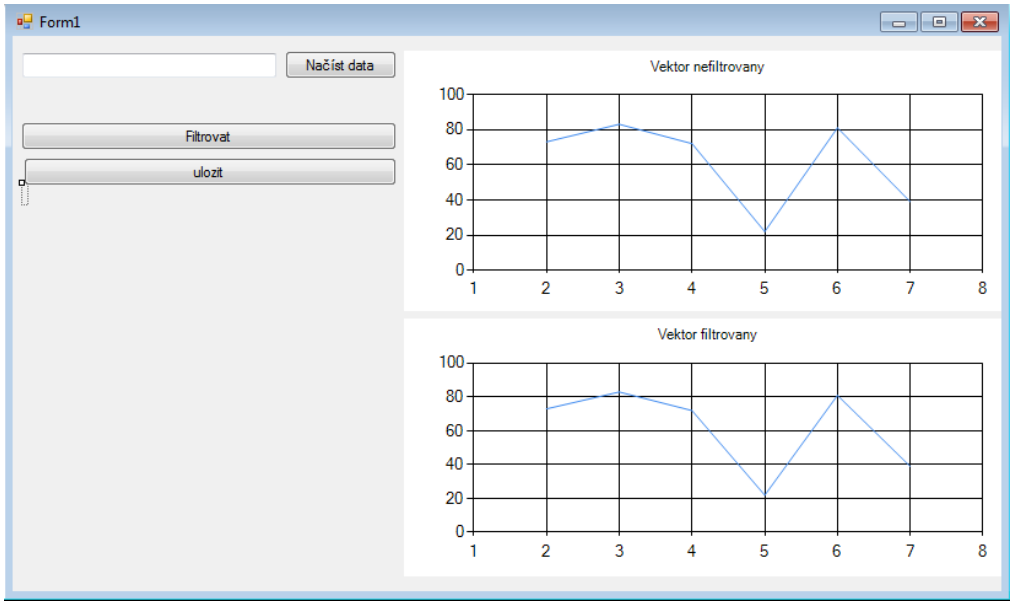
Ze sil působících na motorku lze vyčíst, že vektor zrychlení budu směřovat směrem k zemi a od vektoru v klidu, neboli gravitačního zrychlení, neboli kalibračnímu vektoru se v případě normální jízdy odkloní jen o určitý úhel. Z toho plyne, že pro jednoduché vyhodnocení, zda nastala nehoda, nebo ne, stačí kontrolovat úhel mezi vektorem zrychlení a kalibračním vektorem, a pokud je tento úhel větší než nějaká určená hodnota, bude indikována nehoda. Úhel mezi těmito dvěma vektory byl stanoven na 50°. Tento princip je ovšem narušen schopností motorky dosáhnou velkého zrychlení, nebo naopak zpomalení. Při těchto manévrech může vektor zrychlení přesáhnout oněch 50°. Proto je potřeba do vyhodnocení přidat ještě kontrolu, jestli je vektor daný okamžitým zrychlením přibližně roven gravitačnímu zrychlení.Při zrychlení nebo brždění se vektor zrychlení zvětšuje a mění směr a jeho velikost by se jenom náhodně mohla rovnat gravitačnímu zrychlení. Pro tyto případy je k výše zmíněným dvěma podmínkám ještě dodáno počítadlo, které počítá, kolikrát za sebou byly detekovány podmínky havárie. Pokud dosáhne hodnoty 10, je stav vyhodnocen jako havárie, pokud ale v průběhu počítání nastanou opět normální podmínky, nehoda není detekována.

Tento systém zaručí, že detekce nehody neproběhne při normální jízdě.

5.3.5 Ukládání dat

Ukládání dat je realizováno podobně jako načítání. Použil jsem metody SaveFileDialog a StreamWriter.

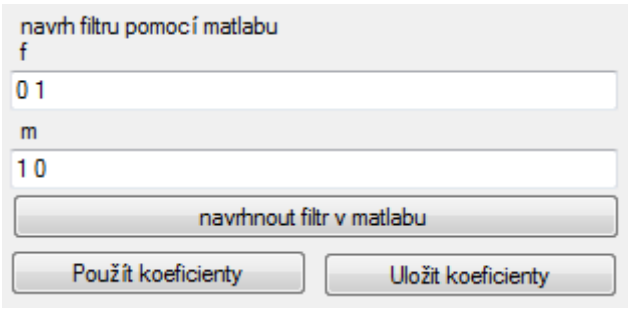
Data se ukládají ve formátu .csv a prozatím ukládám jen vyfiltrovaná data ze všech tří os. Ukládání probíhá do stejné složky, jako byl původní soubor, jen je vytvořen nový soubor se stejným názvem, kde je na začátek připsáno *filtrovana_* pro odlišení od původních dat.



Obr. 5-7 První verze aplikace

5.3.6 Spolupráce s Matlabem

Z důvodu úspory času při návrhu koeficientů filtru jsem do své aplikace přidal možnost nechat Matlab navrhnout koeficienty. Pro interakci s Matlabem jsem musel přidat do aplikace několik prvků Obr. 5-8.



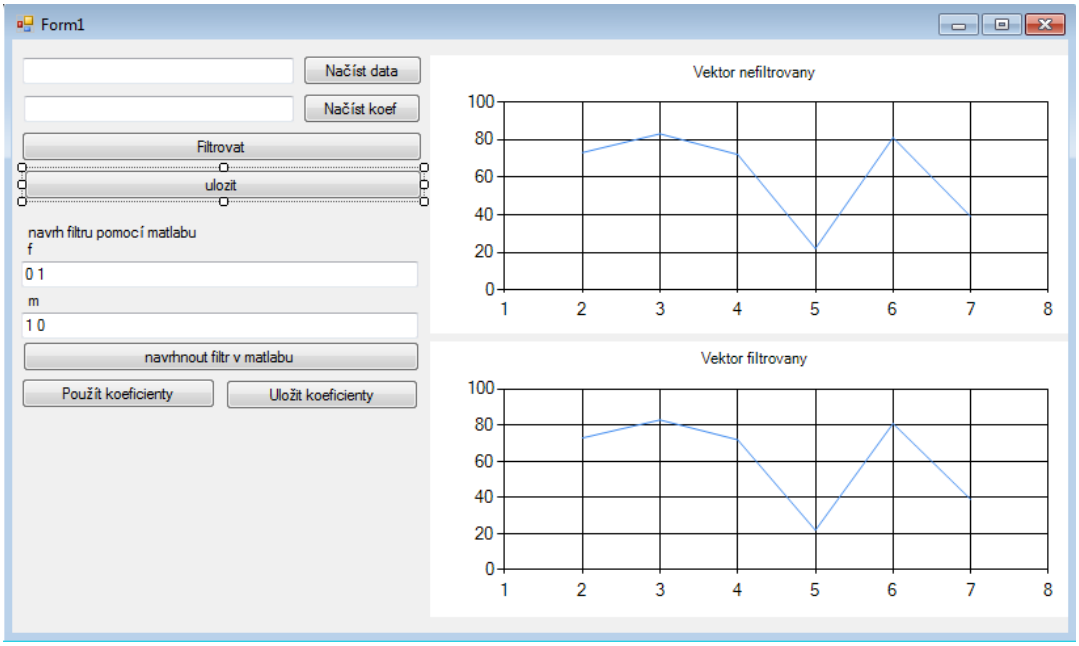
Obr. 5-8 Prvky pro návrh v programu Matlab

Při spojení Matlabu a C# spustil Matlab jako COM Automation Server a C# se choval jako klient. Vytvoření spojení a realizace komunikace byla velice jednoduchá hlavně díky informacím získaným z [9], [10].

Při realizaci spojení jsem měl největší problém se čtením dat z Matlabu, musel jsem se totiž vypořádat s několika překážkami. Data z Matlabu jsem četl jako textový řetězec (string) a pro další zpracování jsem musel konvertovat na číslo double. Matlab ukládal vypočítané hodnoty jak v exponenciálním tvaru, tak ve tvaru desetinném, a proto jsem musel ve své aplikaci umět rozpoznat a převést obě tyto možnosti. Dále byl problém s desetinnou čárkou a tečkou, kde Matlab pracuje s desetinnou tečkou a můj program s desetinnou čárkou. Proto jsem musel přidat při načítání dat také úpravu textových řetězců.

Pro návrh filtru v Matlabu je použita funkce fir2(n, f, m), kde jako *n* je předán řád filtru (v mém případě 8), a *f* a *m* jsou vektory, které specifikují požadovanou odezvu filtru. Vektor *f* obsahuje frekvence, u kterých je vyžadována magnituda uložená ve vektoru *m*. Z toho vyplývá, že oba dva vektory musí mít

stejnou velikost. Hodnoty ve vektoru *f* jsou od nuly do jedné a musí mít vzrůstající charakter. Jednička se rovná Nyquistově frekvenci, což je polovina vzorkovací frekvence, která by měla dle vzorkovacího teorému být vyšší než nejvyšší frekvence ve vzorkovaném signálu.



Obr. 5-9 Druhá verze aplikace

Na Obr. 5-9 můžeme vidět druhou verzi aplikace s částí pro návrh pomocí Matlabu. Dále už tato druhá verze aplikace umožnovala vybrat soubor, ze kterého se načítají koeficienty, tuto funkci jsem do aplikace přidal, aby bylo možné rychle a efektivně testovat více typů filtrů.

5.3.7 Zpracování načtených dat

Pro potřeby vyhodnocení kvality navrhnutého filtru je potřeba před provedením samotné filtrace analyzovat načtená data.

Nejprve je potřeba vyhodnotit příznaky vytvořené v MCU. Příznaky jsou načteny ve vlastním listu a ten je poskytnut funkci, která pouze spočítá, kolik kterých příznaků bylo vyhodnoceno mikrokontrolérem. Toto jsou mé cílové hodnoty, neboť pro účel mé práce jsem považoval filtr v MCU za dostatečně kvalitní.

Poté je potřeba vytvořit příznaky, které by byly vyhodnoceny, kdyby nebyla data filtrována. Z toho důvodu jsem vytvořil několik funkcí, které zajistí vyhodnocení dat a vytvoření příznaků podle systému popsaného v kapitole 5.3.3. Pro vytvoření příznaku je potřeba nejprve spočítat velikost úhlu mezi okamžitým vektorem a vektorem kalibračním. Druhá funkce vytvoří velikost okamžitého vektoru. Výstupy těchto dvou funkcí jsou použity pro funkci *TvorbaPriznaku*, která z těchto dat umí vytvořit dané příznaky, tato funkce musí brát ohled i na pořadí příznaků. Jakmile jsou příznaky vytvořeny, mohou být uloženy do druhého listu. Tyto příznaky jsou mým výchozím bodem, neboť pokud i po použití filtru dojde ke stejnému počtu alarmů, nemá filtrace v daném případě význam.

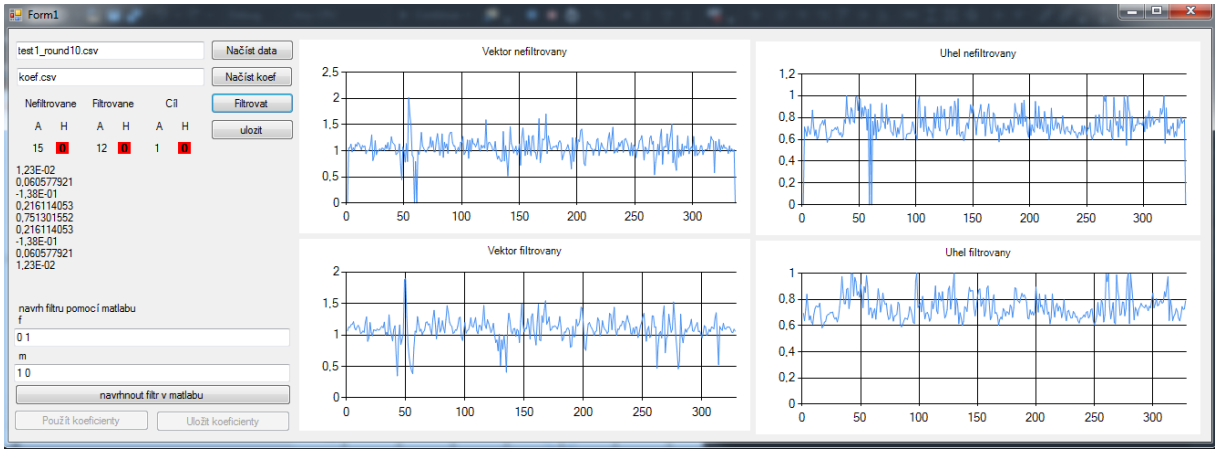
Jakmile jsou příznaky vytvořeny a spočítány mohou být zobrazeny. Pro tento účel je vytvořena malá tabulka, která obsahuje dva sloupce pro každý typ dat. Příznaky mám z tří typů dat nefiltrovaných,

filtrovaných a filtrovaných na MCU. Na Obr. 5-10 můžeme vidět stav po načtení dat, kdy už je možno spočítat a zobrazit příznaky pro nefiltrovaná data. Písmeno A označuje alarmy a písmeno H stav, kdy již nastala havárie. Pro zvýraznění je počet havárii podbarven červeně.

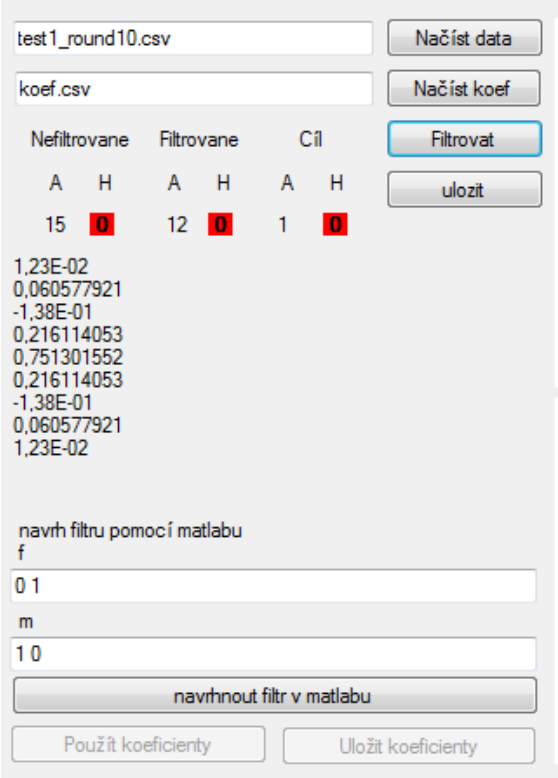
Obr. 5-10 Část aplikace zobrazující příznaky

V tuto chvíli má aplikace uměla načíst koeficienty filtru, načíst data a vyfiltrovat je a pak je zpracovat tak, aby se daly porovnat účinnosti jednotlivých filtrů.

Ovšem při testování jsem zjistil, že bude výhodné, když má aplikace zobrazí načtené koeficienty pro rychlejší představu o typu filtru. Dále jsem zjistil, že graf zobrazující hodnoty načtené z MCU a vyfiltrované nemá moc velkou váhu, neboť na těchto hodnotách je velikost vektoru zkreslená a taky úhel není patrný. Z tohoto důvodu jsem přidal ještě dva grafy tak, aby byla zobrazena velikost vektoru a jeho úhel jak pro filtrovaná data, tak pro data nefiltrovaná. Tyto změny mi pomohly při následném výběru vhodného filtru.



Obr. 5-11 Třetí verze aplikace



Obr. 5-12 Ovládací část třetí verze aplikace

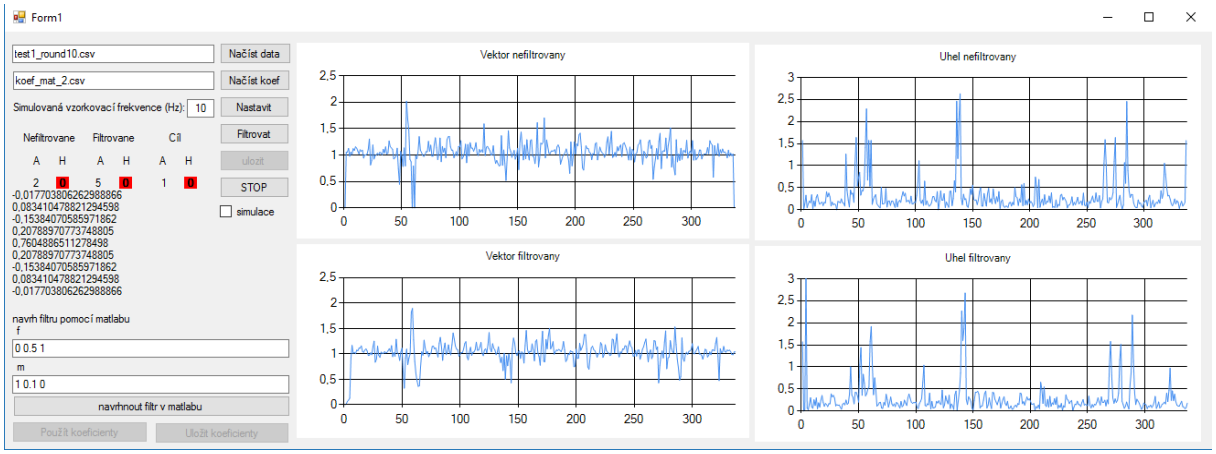
Z Obr. 5-12 je vidět, že aplikace například neumožní uložit koeficienty navrhnuté Matlabem, pokud nejprve neprovedeme jejich samotný návrh.

5.3.8 Filtrace

Nejprve probíhala filtrace pro všechna data v jednom kroku a nebylo možné simulovat chování na MCU. Následně jsem program upravil a filtraci převedl do timeru, kde je možné měnit rychlost filtrace. Po této úpravě je možné provést simulaci, kdy se nastaví timer tak, aby prováděl výpočty stejně často, jako je vzorkovací frekvence. Vyfiltrovaná data jsou také průběžně vyhodnocována a jsou zapisována do grafu a taky jsou vyhodnocovány příznaky. Pro vyhodnocení jsem použil stejné funkce jako pro nefiltrovaná data, jenom jsem jim předložil jiná data.

5.4 Finální verze

Finální verze umožnuje všechny výše popsané funkce a taky umožňuje upravovat rychlost filtrace přímo v aplikaci.



Obr. 5-13 Finální verze aplikace

Na této verzi jsem si mohl opět otestovat všechny verze mých navrhnutých filtru a potvrdit si své závěry, které jsem udělal při průběžném návrhu filtru.

6 Vyhodnocení filtrace

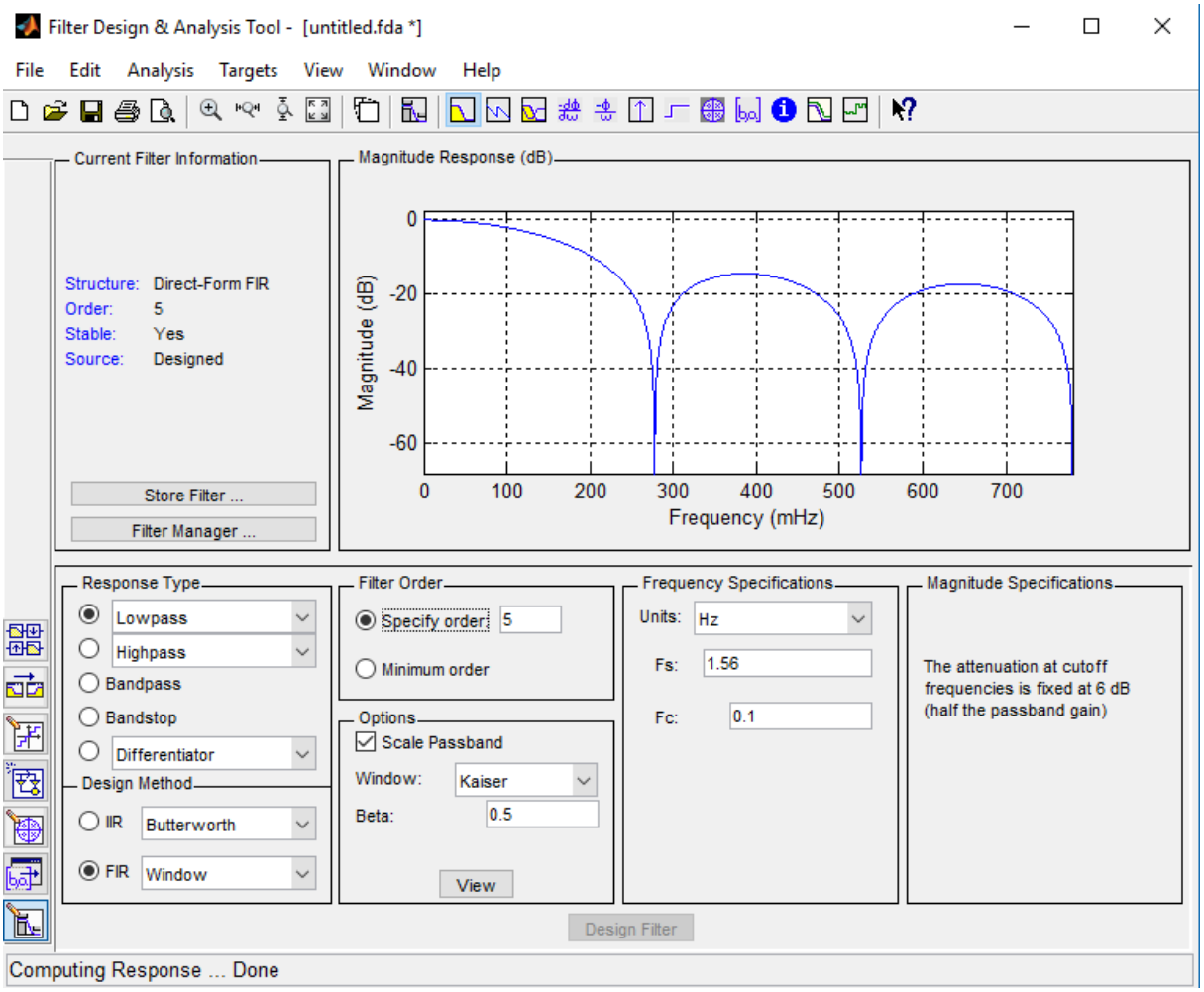
Pro filtrování dat jsem vyzkoušel několik typů filtrů. Pří návrhu filtrů jsem zkoušel měnit mezní frekvenci (fc), ale taky typ návrhového algoritmu. Vyzkoušel jsem několik typů oken a nejlepších výsledků jsem dosáhl s Kaiserovým a Hammingovým oknem. Mimo okenní funkce jsem zkoušel i jiné typy návrhu, které mi nabízel Matlab, ale u některých bylo složitější navrhnout typ filtru, jaký jsem požadoval, a taky jsem nedosáhl tak dobrých výsledků.

Při testování filtrů jsem postupoval tak, že navrhnutý filtr jsem otestoval na 3 sadách dat, a to na datech, kde nastala nehoda, a na dvou různých datech bez nehody. Pokud jsem byl s výsledky spokojen, filtr jsem uložil pro další testování. Při návrzích filtrů jsem často navrhnul filtr, který nebyl schopen detekovat nehodu, i když v datech byla, a proto jsem tyto filtry musel zrušit. Dále jsem také narazil na filtry, které místo aby alarmní stavy odfiltrovaly, tak tyto stavy generovaly. Tomuto jevu jsem se nedokázal úplně vyhnout u žádného filtru, ale pokud se objevoval příliš často, nepovažoval jsem takový filtr za vhodný. Nakonec jsem získal 5 filtrů, které jsem aplikoval na všechna data a vytvořil tabulky výsledků pro tyto filtry. Při porovnání jsem spočítal počet alarmů, tím jsem získal hodnotu, která charakterizuje kvalitu filtru, a filtr s nejmenší hodnotou je nejlepší.

Tab. 6-1 Výsledky filtrace

	Filtr 1	Filtr 2	Filtr 3	Filtr 4	Filtr 5
klidovy_stav	0	0	0	0	0
meas_01	1	0	0	0	0
meas_02	0	0	0	0	0
meas_03	0	0	0	0	0
meas_04	3	1	3	3	3
meas_05	1	2	1	1	1
test1_pad1	2	2	2	3	2
test1_round1	1	0	0	0	0
test1_round2	1	0	0	0	1
test1_round3	2	1	2	1	3
test1_round4	5	2	2	2	2
test1_round5	1	0	1	0	1
test1_round6	0	0	0	0	0
test1_round7	0	0	0	0	0
test1_round8	0	0	0	0	0
test1_round9	1	0	0	0	0
test1_round10	1	0	0	1	0
test2_round1	0	0	0	0	0
test2_round2	0	0	0	0	0
test2_round3	0	0	0	0	0
test2_round4	0	0	1	1	1
test2_round5	0	0	0	0	0
součet:	19	8	12	12	14

V Tab. 6-1, můžeme vidět, že nejlepších výsledků jsem dosáhl s filtrem číslo 2. Jednalo se o filtr 5. řádu, který byl navrhnutý metodou oken. Jako okno bylo použito Kaiserovo okno. Návrh tohoto filtru byl proveden v Matlabu viz Obr. 6-1



Obr. 6-1 Návrh vybraného filtru v FDATool

7 Závěr

Ve své bakalářské práci jsem se zabýval zrychlením a jeho měřením. V oblasti akcelerometrů jsem prošel největší výrobce a jejich produkty. Zjistil jsem, že ne všichni výrobci nabízí MEMS akcelerometry, a ti kteří je nabízejí, se snaží své produkty specializovat jako například společnost STMicroelektronic, která vyrábí akcelerometry s možností implantovat je lidem pod kůži. I přesto mají všichni výrobci v nabídce klasické tříosé akcelerometry určené pro malé zrychlení s přepínatelným rozlišením, úsporným módem, detekcí pádu a dalšími funkcemi.

Pro správný výběr akcelerometru je potřeba znát mnoho parametrů, ale většina akcelerometrů se dá použít pro více aplikací, neboť disponují přepínatelnými rozsahy a různými módy. Klíčovými parametry přesto zůstává citlivost, rychlost snímání, a pokud se jedná o aplikaci, kde je systém napájen baterií, tak taky spotřeba.

Při zpracování dat pomocí filtru v MCU je důležité volit MCU s dostatečnou pamětí a rychlostí výpočtů, většina dnešních MCU má implementovány funkce pro rychlé násobení, což se hodí i pro filtraci.

Pro zpracování dat jsem využil již fungujícího modelu, pro který jsem navrhnul svůj filtr. Pro testy filtrů jsem vytvořil program v jazyce C Sharp, při jeho tvorbě jsem se potýkal se zpracováním dat z mikrokontroléru i z akcelerometru. Nakonec jsem vytvořil program, který umí data zpracovat a vyhodnotit a díky tomu jsem mohl navrhnout vhodný filtr. Tento program dále umí simulovat funkci mikrokontroléru pro zadaná data a detekovat nehodu.

Při návrzích filtrů jsem testoval mnoho parametrů, abych zjistil, jaký mají vliv na výslednou filtraci, a nakonec jsem vybral filtr, který by bylo možné použít i v MCU, neboť se jedná o filtr malého řádu, který by nevyžadoval mnoho výpočtů na MCU pro každý vzorek dat.

Bibliografie

[1] *Omega* [online]. b.r. [cit. 2016-04-08]. Dostupné z: http://www.omega.com/

[2] *Bosch-sensortec* [online]. b.r. [cit. 2016-04-08]. Dostupné z: http://www.bosch-sensortec.com/

[3] *NXP Semiconduktors* [online]. b.r. [cit. 2016-04-08]. Dostupné z: http://www.nxp.com/

[4] *Analog Devices* [online]. b.r. [cit. 2016-04-08]. Dostupné z: http://www.analog.com

[5] *STMikroelektronic* [online]. b.r. [cit. 2016-04-09]. Dostupné z: http://www2.st.com

[6] BIČÁK, Jan, Miloš LAIPERT a Miroslav VLČEK. *Lineární obvody a systémy*. Vyd. 1. Praha: Česká technika - nakladatelství ČVUT, 2007, 204 s. ISBN 9788001036495.

[7] DAVÍDEK, Vratislav. *Analogové a číslicové filtry*. Vyd. 2. Praha: Vydavatelství ČVUT, 2004, vii, 345 s. ISBN 8001030261.

[8] VAN SICKLE, Ted. *Programming microcontrollers in C*. 2nd ed. Eagle Rock, Calif.: LLH Technology Pub., c2001, xvi, 454 p. ISBN 18-787-0757-4.

[9] MALANÍK, Jakub. *Vzájemné využití MATLAB a Microsoft .NET Framework*. Praha, 2013. Bakalářská práce. České Vysoké Učení Technické, Fakulta elektrotechniky, Katedra kybernetiky. Vedoucí práce Ing. Petr Novák, Ph.D.

[10] PODBIELSKI, . Using Matlab from a C# application. In: *CODE PROJECT* [online]. Polsko: Code Project, 2013 [cit. 2016-01-17]. Dostupné z: http://www.codeproject.com/Articles/594636/Using-Matlab-from-a-Csharp-application

Seznam příloh

Vytvořený program Příloha na CD